

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2001-005680

(43)Date of publication of application : 12.01.2001

(51)Int.Cl.

G06F 9/46

G06F 17/60

G06F 19/00

(21)Application number : 11-179520

(71)Applicant : OKI ELECTRIC IND CO LTD

(22)Date of filing : 25.06.1999

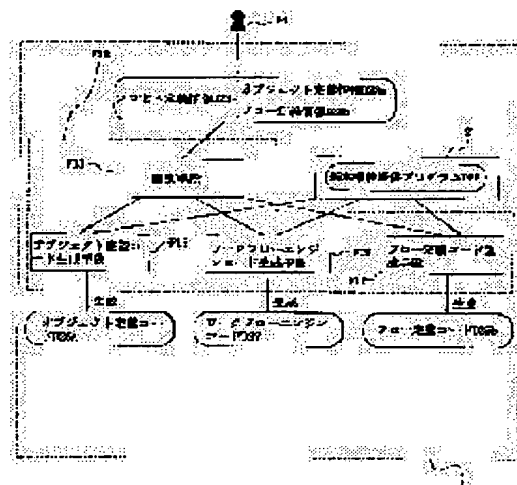
(72)Inventor : OTOGAWA SHINICHI

## (54) WORK FLOW MANAGEMENT SYSTEM, CODE GENERATOR, AND METHOD FOR CHANGING PROCESS OF WORK FLOW PROCESSING

### (57)Abstract:

**PROBLEM TO BE SOLVED:** To easily correct the process of work flow processing.

**SOLUTION:** The process of work flow processing consisting of a series of plural activity elements is defined by inputting process definition data. In the workflow management system 11, a discrimination means F13 separates process definition information D23 inputted by a slip-drafting person 24 into object definition information D23a concerned with the processing contents of each activity and flow definition information D23b concerned with the order of activity elements. An object definition code D25a and a flow definition code D25b are generated from definition information D23a, D23b, respectively. Since definition codes D25a, D25b are independent data of each other, the work flow processing processes to be driven on the basis of these definition codes D25a, D25b can be changed by correcting individual definition codes D25a, D25b.



### LEGAL STATUS

[Date of request for examination]

14.02.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's

## Best Available Copy

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-5680

(P2001-5680A)

(43) 公開日 平成13年1月12日 (2001.1.12)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	テーマコード* (参考)
G 0 6 F 9/46	3 6 0	G 0 6 F 9/46	3 6 0 B 5 B 0 4 9
17/60		15/21	L 5 B 0 9 8
19/00		15/22	N

審査請求 未請求 請求項の数7 O L (全 23 頁)

(21) 出願番号 特願平11-179520

(22) 出願日 平成11年6月25日 (1999.6.25)

(71) 出願人 000000295

沖電気工業株式会社

東京都港区虎ノ門1丁目7番12号

(72) 発明者 乙川 進一

東京都港区虎ノ門1丁目7番12号 沖電気  
工業株式会社内

(74) 代理人 100085419

弁理士 大垣 孝

Fターム(参考) 5B049 AA01 CC32 DD01 EE31 GG07

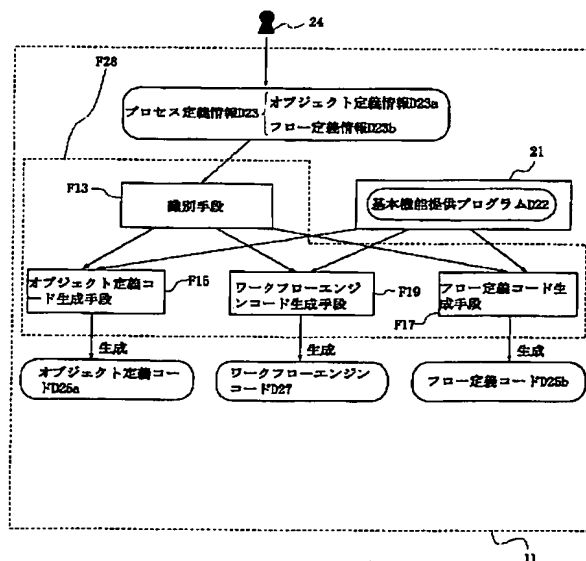
5B098 AA10 GA01 GC01 GC16

(54) 【発明の名称】 ワークフロー管理システム、コードジェネレータおよびワークフロー処理のプロセス変更方法

(57) 【要約】

【課題】 ワークフロー処理のプロセスを容易に修正する。

【解決手段】 プロセス定義データを入力することにより、一連かつ複数のアクティビティからなるワークフロー処理のプロセスは定義される。このワークフロー管理システム11では、起票者24によって入力されたプロセス定義情報D23は、識別手段F13によって、各アクティビティにおける処理内容に関するオブジェクト定義情報D23aおよびアクティビティの順序に関するフロー定義情報D23bに分離される。オブジェクト定義情報およびフロー定義情報からそれぞれオブジェクト定義コードD25aおよびフロー定義コードD25bを生成する。オブジェクト定義コードおよびフロー定義コードは独立したデータであるため、これら定義コードに基づいて動作するワークフロー処理についてのプロセスの変更は、個々の定義コードの修正により行うことができる。



11: ワークフロー管理システム 21: プログラム格納部  
24: 起票者 F28: コードジェネレータ

第1の実施の形態のワークフロー管理システム

## 【特許請求の範囲】

【請求項 1】 複数の計算機を有するネットワークシステムで、一連かつ複数の工程からなるワークフロー処理を、各工程における処理内容およびそれら工程の順序に関するプロセス定義情報に基づいて、各工程毎に行うワークフロー管理システムにおいて、

前記プロセス定義情報から、各工程における処理内容に関するオブジェクト定義情報と、工程の順序に関するフロー定義情報とを識別する識別手段と、

前記ワークフロー処理に用いる基本的な機能を提供する複数のプログラムを格納してあるプログラム格納部と、識別された前記オブジェクト定義情報を解釈して該解釈に応じた前記プログラムを前記プログラム格納部から読み出すことにより、該読み出されたプログラムに基づいて各工程における処理内容に関するコードであるオブジェクト定義コードを生成するオブジェクト定義コード生成手段と、

前記フロー定義情報を解釈して該解釈に応じた前記プログラムを前記プログラム格納部から読み出すことにより、該読み出されたプログラムに基づいて各工程の順序に関するコードであるフロー定義コードを生成するフロー定義コード生成手段と、

前記フロー定義コードおよび前記オブジェクト定義コードを解釈することにより、前記ワークフロー処理を制御するワークフローエンジンとを具えることを特徴とするワークフロー管理システム。

【請求項 2】 請求項 1 に記載のワークフロー管理システムにおいて、

識別された前記オブジェクト定義情報および前記フロー定義情報を解釈して該解釈に応じた前記プログラムを前記プログラム格納部から読み出すことにより、該読み出されたプログラムに基づいてワークフローエンジンコードを生成するワークフローエンジンコード生成手段を更に具え、および、前記ワークフローエンジンコードによって前記ワークフローエンジンを機能させることを特徴とするワークフロー管理システム。

【請求項 3】 請求項 1 に記載のワークフロー管理システムにおいて、

前記オブジェクト定義コードおよび前記フロー定義コードを高水準言語のコードの状態で保存する定義コード保存手段を更に具えることを特徴とするワークフロー管理システム。

【請求項 4】 請求項 1 に記載のワークフロー管理システムにおいて、

前記ワークフローエンジンが上位階層から複数の下位階層に機能を継承する階層構造を有し、かつ、個々の前記下位階層のワークフローエンジンは相互にアクセスする通信機能を有することを特徴とするワークフロー管理システム。

【請求項 5】 請求項 4 に記載のワークフロー管理シ

テムにおいて、

前記フロー定義コードに一般的な宛先名を予め定義する場合、該一般的な宛先名から具体的な宛先名を生成する宛名変換手段を更に具えており、および、

前記宛名変換手段が、前記下位階層のワークフローエンジンと相互にアクセスする通信機能を有することを特徴とするワークフロー管理システム。

【請求項 6】 複数の計算機を有するネットワークシステムで、一連かつ複数の工程からなるワークフロー処理を行う際、各工程における処理内容およびそれら工程の順序に関するプロセス定義情報から、ワークフローエンジンの解釈できるプロセスに関するコードを生成するコードジェネレータであって、

前記プロセス定義情報から、各工程における処理内容に関するオブジェクト定義情報と、工程の順序に関するフロー定義情報とを識別する識別手段と、

前記オブジェクト定義情報から、ワークフローエンジンの解釈できるオブジェクト定義コードを生成するオブジェクト定義コード生成手段と、

前記フロー定義情報から、ワークフローエンジンの解釈できるフロー定義コードを生成するフロー定義コード生成手段とを具えることを特徴とするコードジェネレータ。

【請求項 7】 一連かつ複数の工程からなるワークフロー処理を行う前に、各工程の処理内容および工程の処理順序に関するプロセス定義情報から、各工程の処理内容に関するオブジェクト定義情報と、工程の処理順序に関するフロー定義情報とを識別し、前記オブジェクト定義情報からワークフローエンジンの解釈できるオブジェクト定義コードを生成し、かつ、前記フロー定義情報からワークフローエンジンの解釈できるフロー定義コードを生成するワークフロー処理のプロセスを変更するに当たり、

各工程の処理内容のみを変更する場合にはオブジェクト定義コードのみを修正し、工程の処理順序のみを変更する場合にはフロー定義コードのみを修正することを特徴とするワークフロー処理のプロセス変更方法。

## 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】この発明は、ワークフロー管理システム、ワークフロー管理システムに用いるコードジェネレータおよびワークフロー処理のプロセスの変更方法に関する。

## 【0002】

【従来の技術】従来、ワークフロー管理システムが、複数の端末計算機が接続されたネットワークシステムで、一連かつ複数の工程からなる情報処理すなわちワークフロー処理を効率よく行うために利用されている。ワークフロー処理に係るデータは、パッケージとして各計算機の間を定められた順序で移動し、および、各計算機で処

理される。これらの各計算機を操作して各処理者がデータを参照したり書き換えたりする。このような、ワークフロー管理システムの一例が文献Ⅰ（日経コンピュータ」1997. 9. 1号 第204～217頁）に開示されている。

【0003】図9は、文献Ⅰに開示されているワークフロー管理システムを各機能毎にブロックで分割して概略的に示す図である。図9では、データを角の丸い長方形で示し、機能手段（アプリケーション等）を角張った長方形で示してある。図9に示すように、このワークフロー管理システム201は、ワークフローエンジンA203と、プロセス定義ツールA205と、ワークリストハンドラA207と、モニタリングツールA209とを有する。広く用いられているクライアント／サーバシステムでは、ワークフロー処理を制御するワークフローエンジンA203がワークフローサーバ204で動作しており、各処理者はクライアント計算機を操作して担当の工程（アクティビティ）を実行する。

【0004】プロセス定義ツールA205は、各工程の処理内容と、アクティビティの順序とを定義するための機能モジュールである。プロセス定義ツールA205は、新規にワークフロー処理を開始するとき、プロセス定義データD211を生成する。プロセス定義データD211は、各アクティビティにおける処理内容と、アクティビティの順序とに関するデータを一体のデータとして含む。また、プロセス定義データD211は、ワークフロー処理の終了条件等の制御条件に関するデータを含むことがある。一般的なプロセス定義ツールA205ではグラフィカルユーザインターフェース（GUI）環境が整えられている。そのため、処理者は、マウス操作によってポインタを用いることによりプロセス（アクティビティの順序および各アクティビティでの処理内容）を容易に定義できる。

【0005】ワークフローエンジンA203は、ワークフロー処理を制御する機能モジュールである。具体的に言うと、ワークフローエンジンA203は、プロセス定義データD211を解釈してこの解釈に基づいて処理内容を示すワークアイテムをワークリストD213に追加する。なお、ワークリストD213は、各処理者が処理すべきワークアイテムを格納するデータ領域であり、個々の処理者毎に設けられる。

【0006】各処理者は、ワークリストハンドラA207によって示されるワークアイテムにしたがって、クライアントアプリケーションA215等を用いてアクティビティを実行する。また、人間による処理を必要としないアクティビティについては、ワークフローエンジンA203によって起動された起動アプリケーションA217が、アクティビティを実行する。図9に示すように、これらクライアントアプリケーションA215或いは起動アプリケーションA217はアプリケーションデータ

D219を利用することがある。

【0007】ワークリストハンドラA207は、ワークリストD213を参照して処理すべきワークアイテムを各処理者に示すことにより、各処理者に処理内容に関する指示を与える。その処理者が作業を終了すると、ワークリストハンドラA207はワークリストD213からワークアイテムを削除する。ワークフローエンジンA203は、ワークアイテムの削除によってアクティビティの終了を認識して、次の処理者のワークリストD213にワークアイテムを付加する。

【0008】モニタリングツールA209は、ワークフローエンジンA203を監視することによってワークフロー処理の進行具合を確認する機能モジュールである。モニタリングツールA209によって、監督者或いは各処理者は必要に応じてワークフロー処理の途中経過や終了結果を参照することができる。

【0009】ワークフロー制御データD221は、複数のワークフロー処理を並行して行う場合、各ワークフロー処理を受け持つワークフローエンジンを互いに識別するのに用いられる。また、ワークフロー関連データD223は、ワークフロー処理に関連する様々な情報を含む。ワークフロー関連データD223は、例えば購入物品の価格データ等であり、価格帯によって担当すべき処理者が変化する場合等に用いられる。

【0010】図9に示すように、ワークフローエンジンA203がプロセス定義データD211に基づいて各アクティビティにおける処理を制御することにより、ワークフロー処理は行われる。一般的に言うと、ワークフロー処理は、繰り返し行われる定型業務およびアドホックな非定型業務の二つの業務に分類される。定型業務には出張費伝票、備品購入伝票または予算申請などがあり、非定型業務には資料回覧、技術文書作成またはソフト開発などがある。業務の効率を向上させるためには、これら多種多様な業務に適用できるワークフロー管理システムが必要である。

【0011】典型的に言うと、定型業務のワークフロー処理では、アクティビティの順序および各アクティビティにおける処理内容が殆ど変化しないため、一つのプロセス定義データを用いることが多い。一方、非定型業務のワークフロー処理では、アクティビティの順序および各アクティビティにおける処理内容が一定ではないので、ワークフロー処理を行う毎にプロセス定義データを生成することが多い。

【0012】

【発明が解決しようとする課題】しかしながら、従来のワークフロー管理システムでは、定型業務或いは非定型業務のいずれにおいても、各アクティビティの処理内容或いは処理順序を変更する場合にはプロセス定義データの全体を修正する必要があった。例えば、ルーチンワークである備品購入伝票において決裁者を変更する場合に

は、今まで用いていたプロセス定義データを修正しなければならない。また、アドホックに発生する資料回覧において資料の回覧先を変更したい場合には、資料回覧を開始する前にプロセス定義データを修正しなければならない。

【0013】このようなアクティビティの処理順序の変更を容易に行うため、図9のワークフロー管理システム201では、ワークフローエンジンA203が組織/役割データD225を参照する。組織/役割データD225とは、課長や部長などの組織内の役職等を示す一般的宛先名と、処理者の名前等の具体的宛先名とを関連付けたデータである。この場合、プロセス定義データD211に一般的宛先名を定義しておくだけで、ワークフローエンジンA203は、組織/役割データD225を参照することにより一般的宛先名から具体的宛先名を判断できる。そのため、決裁ルートに沿う役職にある処理者が人事異動等によって他の処理者に変更された場合でも、動的に宛先が認識されるため、決裁ルートの宛先名を変更する必要がなくなる。

【0014】このような組織/役割データの導入によって変更が不必要になる場合はあるが、役職で処理順序を定義しないワークフロー処理では、依然としてアクティビティの処理順序をプロセス定義データから修正する必要があった。

【0015】したがって、定型業務および非定型業務のいずれのワークフロー処理においても、一つのアクティビティにおける処理内容を変更する場合、或いは、アクティビティの処理順序を変更する場合には、プロセス定義データを修正し或いはプロセス定義データを新たに作成する。このように、従来のワークフロー処理では、プロセスに関する変更はプロセス定義データの再定義を伴うため非能率的であり、容易に行うことができなかった(第1の問題点)。

【0016】また、通常、新たにワークフロー処理を開始するときには、プロセス定義データを新規に作成する。このとき、ワークフロー処理を適用する業務の種類が多岐に渡る場合には、必然的にプロセス定義データも様々な内容を有することとなる。ワークフローエンジンは、それらのプロセス定義データを解釈してかつ実行できなければならないから、多種のワークフロー処理を実行するための多くの基本的な機能を予め具備している必要がある。したがって、ワークフローエンジンの構成が複雑となり、ワークフローエンジンとして機能するワークフローエンジンコードの占有メモリが必然的に大きくなる(第2の問題点)。

【0017】そのため、少なくとも第1の問題点を解決でき、好ましくは第2の問題点も解決できるワークフロー管理システムが望まれていた。

【0018】

【課題を解決するための手段】したがって、この出願に

係る発明のワークフロー管理システムは、複数の計算機を有するネットワークシステムで、一連かつ複数の工程からなるワークフロー処理を、各工程における処理内容およびそれら工程の順序に関するプロセス定義情報に基づいて、各工程ごとに行うワークフロー管理システムにおいて、前述のプロセス定義情報から、各工程における処理内容に関するオブジェクト定義情報と、工程の順序に関するフロー定義情報とを識別する識別手段と、前述のワークフロー処理に用いる基本的な機能を提供する複数のプログラムを格納してあるプログラム格納部と、前述のオブジェクト定義情報を解釈してこの解釈に応じたプログラムをプログラム格納部から読み出し、かつ、この読み出されたプログラムに基づいて各工程における処理内容に関するコードであるオブジェクト定義コードを生成するオブジェクト定義コード生成手段と、前述のフロー定義情報を解釈してこの解釈に応じたプログラムをプログラム格納部から読み出し、かつ、この読み出された各工程の順序に関するコードであるフロー定義コードを生成するフロー定義コード生成手段と、前述のフロー定義コードおよび前述のオブジェクト定義コードを解釈することにより、ワークフロー処理を制御するワークフローエンジンとを具える。

【0019】このワークフロー管理システムの構成によれば、ワークフロー処理のプロセスを定義をする際、起票者がワークフロー処理を開始させるためにプロセスを定義する。起票者が入力したプロセス定義情報は、識別手段によってフロー定義に関する部分およびオブジェクト定義に関する部分に分離される。この分離の後、前者の部分はフロー定義コード生成手段によってコード化されてフロー定義コードとして生成され、および、後者の部分はオブジェクト定義コード生成手段によってコード化されてオブジェクト定義コードとして生成される。よって、ワークフロー処理を行うとき、プロセス(各アクティビティにおける処理内容およびアクティビティの順序)を変更する場合であっても、プロセス定義データそのものを修正する必要がない。言い換えれば、各アクティビティにおける処理内容のみを変更したい場合にはオブジェクト定義コードのみを修正すれば良く、アクティビティの順序(アクティビティを行う処理者の変更)を変更したい場合にはフロー定義コードのみを修正すればよい。

【0020】また、この発明の実施に当たり、より好適には、前述のオブジェクト定義情報および前述のフロー定義情報を解釈してこの解釈に応じたプログラムをプログラム格納部から読み出すことにより、この読み出されたプログラムに基づいてワークフローエンジンコードを生成するワークフローエンジンコード生成手段を更に具え、および、このワークフローエンジンコードによって前述のワークフローエンジンコードを機能させるのが良い。

【0021】それにより、フロー定義情報およびオブジェクト定義情報に定義された機能を実現するための必要最低限の構成のみを有するワークフローエンジンコードを生成できる。よって、ワークフローエンジンコードの占有メモリを小さくできる。なお、ワークフローエンジンコードとは、CPUで実行されることにより仮想的にCPUにワークフローエンジンを存在させかつ機能させるプログラムコードを意味する。

【0022】また、この発明の実施に当たり、好適には、前述のオブジェクト定義コードおよび前述のフロー定義コードを高水準言語のコードの状態で保存する定義コード保存手段を更に具えるのが良い。それにより、フロー定義或いはオブジェクト定義についての修正をするとき、修正する者は、各定義コードを理解しながらコードを追加したり削除したり或いは修正したりすることができる。

【0023】また、この発明の実施に当たり、好適には、前述のワークフローエンジンは上位階層から複数の下位階層に機能を継承する階層構造を有し、かつ、個々の下位階層のワークフローエンジンは相互にアクセスする通信機能を有するのがよい。それにより、下位階層のワークフローエンジンをネットワークシステム内のホスト計算機に分散できる。このように、ワークフローエンジンに階層構造を与え、個々の下位階層のワークフローエンジンに通信機能を設けると、下位下層のワークフローエンジンは、ネットワークシステム内の任意のホスト計算機に分散して存在できる。よって、負荷分散によるネットワークシステムの効率および安定性が実現できる。

【0024】また、上述のように下位階層のワークフローエンジンを分散させてなるワークフロー管理システムでは、前述のフロー定義コードに一般的な宛先名を予め定義する場合、一般的宛先名から具体的宛先名を生成する宛名変換手段を更に具えていて、この宛名変換手段が、ワークフローエンジンと相互にアクセスする通信機能を有するのが望ましい。

【0025】従来、宛名変換を行うには、宛名変換手段はワークフローエンジンの存在するホスト計算機と同一のホスト計算機に設ける必要があった。しかしながら、宛名変換手段に通信機能を設けたため、上述のように下位階層のワークフローエンジンを分散させた場合であっても、宛名変換手段は下位階層のワークフローエンジンと相互にアクセスできる。よって、宛名変換手段を下位階層のワークフローエンジンと別のホスト計算機に設けることができる。

【0026】また、この出願に係る発明のコードジェネレータは、複数の計算機を有するネットワークシステムで、一連かつ複数の工程からなるワークフロー処理を行う際、各工程における処理内容およびそれら工程の順序に関するプロセス定義情報から、ワークフローエンジン

の解釈できるコードを生成するコードジェネレータであって、前述のプロセス定義情報から、各工程における処理内容に関するオブジェクト定義情報と、工程の順序に関するフロー定義情報とを識別する識別手段と、前述のオブジェクト定義情報からワークフローエンジンの解釈できるオブジェクト定義コードを生成するオブジェクト定義コード生成手段と、前述のフロー定義情報からワークフローエンジンの解釈できるフロー定義コードを生成するフロー定義コード生成手段とを具える。

【0027】このコードジェネレータの構成によれば、識別手段が、プロセス定義情報をオブジェクト定義情報およびフロー定義情報に分離させたのち、オブジェクト定義情報からオブジェクト定義コードを生成しかつフロー定義情報からフロー定義コードを生成する。そのため、各工程における処理内容のみを変更したい場合にはオブジェクト定義コードを修正し、また、工程の処理順序のみを変更したい場合にはフロー定義コードを修正すればよい。よって、プロセスを能率的にかつ容易に変更することができる。なお、オブジェクト定義コードおよびフロー定義コードは、例えば、高水準言語としてのJava言語或いはC++のソースプログラムコードすなわち、オブジェクト定義ソースプログラムコードおよびフロー定義ソースプログラムコードとして生成される。このとき、これらオブジェクト定義コードおよびフロー定義コードは、コンパイルされると、ワークフローエンジンが解釈できる実行型プログラムコードとなる。

【0028】また、この出願に係るワークフロー処理のプロセスの変更方法の発明では、一連かつ複数の工程からなるワークフロー処理を行う前に、各工程の処理内容および工程の処理順序に関するプロセス定義情報から、各工程の処理内容に関するオブジェクト定義情報と、工程の処理順序に関するフロー定義情報とを識別し、オブジェクト定義情報からワークフローエンジンの解釈できるオブジェクト定義コードを生成し、かつ、フロー定義情報からワークフローエンジンの解釈できるフロー定義コードを生成するワークフロー処理について、当該ワークフロー処理のプロセスを変更するに当たり、各工程の処理内容のみを変更する場合にはオブジェクト定義コードのみを修正し、工程の処理順序のみを変更する場合にはフロー定義コードのみを修正する。

【0029】このプロセスの変更方法によれば、各工程における処理内容のみを変更したい場合若しくは工程の処理順序のみを変更したい場合のいずれにも、互いに独立に存在するオブジェクト定義コード或いはフロー定義コードの一方を修正することにより対応できる。よって、プロセスを能率的にかつ容易に変更できる。

【0030】

【発明の実施の形態】以下、図を参照して、この発明のワークフロー管理システムの実施の形態につき説明する。なお、この説明に用いる各ブロック図は、これら発

明を理解できる程度に各構成成分をブロックで表して概略的な機能上の相互関係を示しているに過ぎない。また、各図において同様な構成成分については、同一の番号を付して示し、その重複する説明を省略することがある。また、各図で角張った長方形は機能単位を示し、角の丸い長方形はデータを示す。

【0031】（第1の実施の形態）ワークフロー管理システムでは、ワークフロー処理のプロセスを定義したのち当該ワークフロー処理を実行する必要がある。そのため、第1の実施の形態では、ワークフロー管理システムのプロセスを定義する形態につき説明し、次にワークフロー管理システムのワークフロー処理を実行する形態につき説明し、続いてワークフロー処理のプロセスを変更する方法につき説明する。

【0032】図1は、第1の実施の形態のワークフロー管理システムについて、主要な機能単位の相互関係を概略的に示すブロック図である。ただし、図1中の各機能単位は、ハードウェア資源としての中央処理装置がその機能単位に対応するプログラムを実行することにより、仮想的に中央処理装置に存在する。以下では、中央処理装置に仮想的に実現される機能単位を角張った長方形、記憶装置のデータ領域に形成されるデータ情報を丸みを帯びた長方形で示してある。また、図2は、図1の各機能単位を1台のホスト計算機で動作させた場合を例として、各機能単位によるハードウェア資源の利用形態を概略的に示す図である。

【0033】以下、図1および図2を参照して、実施の形態のワークフロー管理システムのプロセスを定義する形態につき説明する。ただし、実施の形態のワークフロー管理システムでは特にワークフローエンジンコード生成手段が設けてある。

【0034】周知のごとく、ワークフロー管理システムは、複数のホスト計算機を有するネットワークシステムで、一連かつ複数のアクティビティからなるワークフロー処理を各アクティビティ毎に行うことにより、ワークフロー処理を実行する。

【0035】図1に示す第1の実施の形態のワークフロー管理システム11は、識別手段F13と、オブジェクト定義コード生成手段F15と、フロー定義コード生成手段F17と、ワークフローエンジンコード生成手段F19と、プログラム格納部21とを有する。なお、ここではこれら各機能単位F13、F15、F17およびF19は、図2のホスト計算機h1の中央処理装置（CPU）c1で仮想的な機能手段として動作するように構成されている。

【0036】図2に示すホスト計算機h1は、ハードディスクや光ディスク等の記憶装置m1と、CPUc1と、マンマシンインターフェース用の入力装置i1とを有する。CPUc1には、これら記憶装置m1、入力装置i1、上述の各機能単位F13、F15、F17およ

びF19その他の動作のタイミングや始動或いは停止を含む所要の制御を行うための制御部を図示せずも含んでいる。なお、ホスト計算機h1の記憶装置m1には、識別手段F13の機能を実現させる識別プログラムD13と、オブジェクト定義コード生成手段F15の機能を実現させるオブジェクト定義コード生成プログラムD15と、フロー定義コード生成手段F17の機能を実現させるフロー定義コード生成プログラムD17と、ワークフローエンジンコード生成手段F19の機能を実現させるワークフローエンジンコード生成プログラムD19とが格納されている。

【0037】先ず、ワークフロー処理のプロセスを定義するに当たり、起票者24は、ホスト計算機h1の入力装置i1を操作して記憶装置m1にプロセス定義情報D23を格納する。このとき、起票者24は、ホスト計算機h1の入力装置i1から例えばキーボードからのスク립ト入力或いはポインタデバイスからのGUI操作によって、プロセス定義情報D23を入力する。これにより、プロセス定義情報D23がホスト計算機h1の記憶装置m1に格納される。ただし、プロセス定義情報D23は、各アクティビティの処理内容に関するデータすなわちオブジェクト定義情報D23aと、アクティビティの順序に関するデータすなわちフロー定義情報D23bとを含む。

【0038】プロセス定義情報D23が入力されると、識別手段F13は、記憶装置m1に格納されたプロセス定義情報D23を読み出してきて、このプロセス定義情報D23からオブジェクト定義情報D23aおよびフロー定義情報D23bを互いに識別する。しかる後、識別されたオブジェクト定義情報D23aおよびフロー定義情報D23bを記憶装置m1に格納する。なお、識別手段F13は、記憶装置m1に格納されている識別プログラムD13をCPUc1が読み出しかつ実行することにより、仮想的な機能単位としてCPUc1で動作する。

【0039】プログラム格納部21は、ワークフロー処理に用いる基本的な機能を提供する複数の基本機能提供プログラムD22を格納している。なお、プログラム格納部21は、例えばホスト計算機h1の記憶装置m1内のデータ領域に形成される。

【0040】例えば、この基本機能提供プログラムD22は、ワークフロー処理に用いる基本的な機能を各機能毎に区分された複数のプログラムからなる。このように、基本的な機能が各機能毎に区分されていると、不必要な機能を読み出すことなく必要な機能だけを読み出すことができる。したがって、基本機能提供プログラムD22から生成されるオブジェクト定義コードD25a、フロー定義コードD25bおよびワークフローエンジンコードD27の占有メモリを小さくできる。

【0041】オブジェクト定義コード生成手段F15は、オブジェクト定義情報D23aを解釈してこの解釈

に応じた基本機能提供プログラムD 2 2をプログラム格納部2 1から読み出すことにより、読み出された基本機能提供プログラムD 2 2に基づいてオブジェクト定義コードD 2 5 aを生成する。このとき、具体的にはオブジェクト定義コード生成手段F 1 5はホスト計算機h 1内で次のように機能する。

【0 0 4 2】すなわち、ホスト計算機h 1のCPU c 1が記憶装置m 1に格納されているオブジェクト定義コード生成プログラムD 1 5を読み出してかつ実行する。それにより、オブジェクト定義コード生成手段F 1 5が起動する。オブジェクト定義コード生成手段F 1 5は、記憶装置m 1からオブジェクト定義情報D 2 3 aを読み出すと共に、オブジェクト定義情報D 2 3 aに基づいてオブジェクト定義コードD 2 5 aの生成に必要な基本機能提供プログラムD 2 2かどうかを判断して当該基本機能提供プログラムD 2 2を読み出す。しかるのち、オブジェクト定義コード生成手段F 1 5は、読み出された基本機能提供プログラムD 2 2およびオブジェクト定義情報D 2 3 aを例えば組み合わせるオブジェクト定義コードD 2 5 aを生成する。例えば、このときオブジェクト定義コード生成手段F 1 5は、複数の基本機能提供プログラムD 2 2を読み出してオブジェクト定義コードD 2 5 aを生成しても良い。なお、生成されたオブジェクト定義コードD 2 5 aは、少なくとも各アクティビティにおける共通の処理内容を定義した基本ソースプログラムを含むソースプログラムとして記憶装置m 1に格納される。

【0 0 4 3】フロー定義コード生成手段F 1 7は、フロー定義情報D 2 3 bを解釈してこの解釈に応じた基本機能提供プログラムD 2 2をプログラム格納手段2 1から読み出すことにより、読み出された基本機能提供プログラムD 2 2に基づいてフロー定義コードD 2 5 bを生成する。このとき、具体的にはフロー定義コード生成手段F 1 7はホスト計算機h 1内で次のように機能する。

【0 0 4 4】すなわち、ホスト計算機h 1のCPU c 1が記憶装置m 1に格納されているフロー定義コード生成プログラムD 1 7を読み出してかつ実行する。それにより、フロー定義コード生成手段F 1 7が起動する。フロー定義コード生成手段F 1 7は、記憶装置m 1からフロー定義情報D 2 3 bを読み出すと共に、フロー定義情報D 2 3 bに基づいてフロー定義コードD 2 5 bの生成に必要な基本機能提供プログラムD 2 2かどうかを判断して当該基本機能提供プログラムD 2 2を読み出す。しかるのち、フロー定義コード生成手段F 1 7は、読み出された基本機能提供プログラムD 2 2およびフロー定義情報D 2 3 bを例えば組み合わせるフロー定義コードD 2 5 bを生成する。例えば、このときフロー定義コード生成手段F 1 7は、複数の基本機能提供プログラムD 2 2を読み出してフロー定義コードD 2 5 bを生成しても良い。なお、生成されたフロー定義コードD 2 5 bは、少

なくともアクティビティの順序を定義したソースプログラムを含むソースプログラムとして記憶装置m 1に格納される。

【0 0 4 5】ワークフローエンジンコード生成手段F 1 9は、識別されたオブジェクト定義情報D 2 3 aおよびフロー定義情報D 2 3 bを解釈してこの解釈に応じた基本機能提供プログラムD 2 2をプログラム格納部2 1から読み出すことにより、読み出された基本機能提供プログラムD 2 2に基づいてワークフローエンジンコードD 2 7を生成する。このとき、具体的にはワークフローエンジンコード生成手段F 1 9はホスト計算機h 1内で次のように機能する。

【0 0 4 6】すなわち、ホスト計算機h 1のCPU c 1が記憶装置m 1に格納されているワークフローエンジンコード生成プログラムD 1 9を読み出してかつ実行することにより、ワークフローエンジンコード生成手段F 1 9が起動する。起動したワークフローエンジンコード生成手段F 1 9は、記憶装置m 1からオブジェクト定義情報D 2 3 aおよびフロー定義情報D 2 3 bを読み出すと共に、これらの定義情報D 2 3 aおよびD 2 3 bに基づいてワークフローエンジンコードD 2 7の生成に必要な基本機能提供プログラムD 2 2かどうかを判断して当該基本機能提供プログラムD 2 2を読み出す。しかるのち、ワークフローエンジンコード生成手段F 1 9は、読み出された基本機能提供プログラムD 2 2からワークフローエンジンコードD 2 7を生成する。例えば、このときワークフローエンジンコード生成手段F 1 9は、複数の基本機能提供プログラムD 2 2を組み合わせるワークフローエンジンコードD 2 7を生成する。なお、生成されたワークフローエンジンコードD 2 7は、オブジェクト定義コードD 1 5およびフロー定義コードD 1 7を解釈（ただし、Javaプログラミングにおける「参照」の意味を含む。）してかつこの解釈に基づいてワークフロー処理を制御するためのソースプログラムとして記憶装置m 1に格納される。

【0 0 4 7】また、実際にワークフロー処理を行う際には、例えばホスト計算機h 1のCPU c 1が記憶装置m 1に格納されているワークフローエンジンコードD 2 7を読み出してかつ実行することにより、一つの機能単位としてのワークフローエンジンが起動してワークフロー処理の制御動作を行う。

【0 0 4 8】なお、識別手段F 1 3、オブジェクト定義コード生成手段F 1 5およびフロー定義コード生成手段F 1 7は、コードジェネレータF 2 8という一つの機能単位として認識できる。コードジェネレータF 2 8は、複数のホスト計算機を有するネットワークシステムで、一連かつ複数のアクティビティからなるワークフロー処理を行う際、各アクティビティにおける処理内容およびそれらアクティビティの順序に関するプロセス定義情報から、ワークフローエンジンの解釈できるプロセスに関



するコードを生成する。特に、このコードジェネレータ F 2 8 は、プロセス定義情報 D 2 3 から、オブジェクト定義コード D 2 5 a およびフロー定義コード D 2 5 b を区別してそれぞれ生成する。そのため、一般的なワークフロー管理システムについて、そのワークフローエンジンがオブジェクト定義コード D 2 5 a およびフロー定義コード D 2 5 b を解釈できる構成であれば、このコードジェネレータ F 2 8 の部分のみを適用できる。

【0049】ここで、具体的なワークフロー処理を実行する場合を例として、以上説明した各機能単位につき更に詳細に説明する。

【0050】なお、ここではワークフロー管理システムを、オブジェクト指向プログラミング（OOP）を用いて構築する。周知の如く、J a v a 言語や C + + 言語等は、OOP に適している。OOP を用いると、上述したワークフローエンジンについて、基本的な機能のみを有する上位階層（スーパークラス）から具体的かつ個別な機能を有する下位階層（サブクラス）までを含む、階層構造を有するワークフローエンジンとして定義できる。このとき、オブジェクト定義コードおよびフロー定義コードは、ワークフローエンジンに参照されるオブジェクトとなる。

【0051】図3は、図1のワークフロー管理システムの具体的な機能構成を概略的に示した図であって、図1に示すワークフローエンジンに代えてクラス構造すなわち階層構造を有するワークフローエンジンを用いる場合のワークフロー管理システムについて、その主要な機能単位の相互関係を概略的に示すブロック図である。なお、図3の各機能単位は、ハードウェア資源としての C P U がその機能単位に対応するプログラムを実行することにより、仮想的に C P U に存在する機能単位である。ただし、便宜上、図3ではワークフローエンジンサブクラス F 2 7 a および F 2 7 b は、それぞれホスト計算機 h 2 および h 3 の各 C P U で動作する仮想的な機能単位として示してある。

【0052】図4は、図3のワークフロー管理システムの各機能単位を動作させるネットワークシステムの一例を示す図である。図4には3つのホスト計算機 h 1、h 2 および h 3 が接続されたネットワークシステムが示されている。図4に示すように、各ホスト計算機 h 1、h 2 および h 3 は、C P U c 1、c 2 および c 3 と、ハードディスクや光ディスク等の記憶装置 m 1、m 2 および m 3 と、マンマシンインターフェース用の入力装置 i 1、i 2 および i 3 とを具える。また、図示せず、各ホスト計算機 h 1、h 2 および h 3 は一般に表示装置等を有する。既に説明したように C P U c 1、c 2 および c 3 は、各機能単位を駆動するための制御部を有している。また、C P U c 1、c 2 および c 3 には各ハードウェア資源を駆動するドライバソフトや一般的な O S ソフト等が動作していても良い。なお、図4は、ワークフ

ー管理システムの各構成が存在できる計算機の一例を示しているに過ぎない。

【0053】ここで具体的なワークフロー処理の例として、2つのアクティビティからなるワークフロー処理を実施する。すなわち、図3および図4に示すように、このワークフロー処理では、先ず起票者 2 4 がホスト計算機 h 1 を操作してワークフロー処理のプロセスを定義し、次に処理者 3 1 a がホスト計算機 h 2 を操作してアクティビティを実行し、その後、処理者 3 1 b がホスト計算機 h 3 を操作してアクティビティを実行する。

【0054】図3および図4に示すワークフローエンジンは、上位階層から下位階層までの複数の階層構造を有する。すなわち、このワークフローエンジンを機能させるコードは、上位階層のワークフローエンジンコード（ワークフローエンジンスーパークラスコード）D 2 7 s、下位階層のワークフローエンジンコード（ワークフローエンジンサブクラスコード）D 2 7 a および D 2 7 b という2層の階層構造を有している。

【0055】ただし、ここではワークフローエンジンコード生成手段 F 1 9 は、ソースプログラムコードであるワークフローエンジンサブクラスコード D 2 7 a および D 2 7 b を生成する。なお、このワークフローエンジンサブクラスコード D 2 7 a は、ワークフローエンジンサブクラス F 2 7 a を機能させ、ワークフローエンジンサブクラスコード D 2 7 b は、ワークフローエンジンサブクラス F 2 7 b を機能させる。

【0056】このように階層構造を有するワークフローエンジンを用いる場合、下位階層のワークフローエンジンコードのみを生成するのが好ましい。

【0057】周知の如く、ワークフローエンジンサブクラス F 2 7 a および F 2 7 b は、ワークフローエンジンスーパークラス（図中では機能単位ではなくコードとして示してある。）D 2 7 s から機能的な継承（インヘリタンス）を行うことができる。したがって、ワークフローエンジンスーパークラスコード D 2 7 s で基本機能を定義しておけば、ワークフローエンジンサブクラス F 2 7 a および F 2 7 b では、当該基本機能を継承できる。そのため、ワークフローエンジンサブクラスコード D 2 7 a および D 2 7 b では、当該基本機能と異なる機能のみを定義すればよい。

【0058】このように、ワークフローエンジンコード生成手段 F 1 9 は、OOP に基づいて、基本機能を定義した上位階層のワークフローエンジンコード D 2 7 s から機能的な継承をする下位階層のワークフローエンジンコード D 2 7 a および D 2 7 b であって、かつ、具体的な固有の機能を定義した下位階層のワークフローエンジンコード D 2 7 a および D 2 7 b を生成するのが好ましい。それにより、ワークフローエンジンとして機能させる全体のソースプログラムコードの記述を簡略化できるため、ワークフローエンジンコードを容易に生成でき

る。それと同時に、ワークフローエンジンコードの記憶装置内の占有メモリを低減できる。なお、図3に示すように、ワークフローエンジンスーパークラスコードD27s等を含むクラスライブラリD52は、例えばプログラム格納部21に設けておく。

【0059】また、スーパークラスからサブクラスに渡る階層構造を有するワークフローエンジンコードについて、ワークフローエンジンサブクラスコードD27aおよびD27bに通信機能を設定して互いにアクセスできるようにしておけば、ワークフローエンジンサブクラスD27aおよびD27bは、それぞれネットワークシステム内の任意のホスト計算機に分散して存在できる。よって、負荷分散によるネットワークシステムの効率および安定性の向上が実現できる。

【0060】なお、ここでは、図4に示すように、ホスト計算機h1にて生成されたワークフローエンジンサブクラスコードD27aおよびD27bは、それぞれホスト計算機h2およびh3に分散している。このように、図4のワークフローエンジンサブクラスコードD27aはホスト計算機h2で動作して図3に示すワークフローエンジンサブクラスF27aとして機能し、図4のワークフローエンジンサブクラスコードD27bはホスト計算機h3で動作して図3に示すワークフローエンジンサブクラスF27bとして機能する。図3および図4に示すように、処理者31aは、ホスト計算機h2を操作することによりアクティビティを行い。その後、処理者31bは、ホスト計算機h3を操作することにより別のアクティビティを行う。このように、ワークフローエンジンコードを生成するとき、各処理者のホスト計算機毎に、互いに異なるワークフローエンジンサブクラスを定義するのが良い。それにより、容易に処理者の特定ができるため、ワークフロー処理のプロセスを定義することが容易となる。

【0061】ここでは、上述のごとく処理者毎に下位階層のワークフローエンジンコードを生成するため、次のようなプロセス定義情報を生成する。すなわち、図3では、前述したプロセス定義情報として例えばプロセス定義スクリプトD43を用いる。このとき、起票者24は例えばホスト計算機h1の入力装置i1を操作してプロセス定義スクリプトD43を入力する。このプロセス定義スクリプトD43は、オブジェクト定義スクリプトD43aおよびフロー定義スクリプトD43bを含む。

【0062】このとき、ホスト計算機h1のCPUc1にOSソフトやエディタソフトを動作させておくと、記憶装置m1にはデータとしてのプロセス定義スクリプトD43が格納される。なお、このプロセス定義スクリプトD43は、ワークフロー処理のプロセス定義に関する情報を簡単な内部コードで記述したデータである。ただし、ここでいう内部コードとはスクリプト識別手段F45が認識できるコードである。

【0063】図5は、具体的なワークフロー処理に対するプロセス定義スクリプトの一例である。図5に示すプロセス定義スクリプトは、オブジェクト定義スクリプトD43aおよびフロー定義スクリプトD43bを含む。

【0064】また、これらのスクリプトD43aおよびD43bを相互に識別するため、各スクリプトD43aおよびD43bには異なる識別子が付加されている。図5に示す例では、識別子engines および識別子flowがフロー定義スクリプトD43bを示し、識別子doがオブジェクト定義スクリプトD43aを示す。

【0065】識別子engines はワークフローエンジンサブクラス名を指定する。ここでは、各処理者31aおよび31b毎に異なるワークフローエンジンサブクラスコードが生成されているため、ワークフローエンジンサブクラス名によって各アクティビティを実行する処理者31aおよび31bを定義している。例えば、図5に示すように、処理者31aに対してはワークフローエンジンサブクラスF27a (loanTeller) を定義し、処理者31bに対してはワークフローエンジンサブクラスF27b (loanDecision) を定義している。

【0066】識別子flowは、ワークフローエンジンサブクラス名を順次に指定することにより、アクティビティの処理順序を定義する。すなわち図3では、ワークフローエンジンサブクラスF27a (loanTeller) およびF27b (loanDecision) の順に、アクティビティが実行される。

【0067】識別子doは、各アクティビティにおける処理内容を定義する。すなわち図5に示すように、ワークフローエンジンサブクラスF27a (loanTeller) ではメソッドstart という処理内容が定義され、ワークフローエンジンサブクラスF27b (loanDecision) ではメソッドdecideLoanという処理内容が定義される。

【0068】図3に示すように、このようなプロセス定義スクリプトD43は、ホスト計算機h1のCPUc1で動作するスクリプト識別手段F45に読み込まれる。このとき、スクリプト識別手段F45は、プロセス定義スクリプトD43に含まれる識別子に基づいて、プロセス定義スクリプトD43をオブジェクト定義スクリプトD43aおよびフロー定義スクリプトD43bに分離する。その後、スクリプト識別手段F45は、ホスト計算機h1に分離したオブジェクト定義スクリプトD43aおよびフロー定義スクリプトD43bを記憶装置m1に格納する。なお、スクリプト識別手段F45は、図4に示すホスト計算機h1の記憶装置m1に格納されているスクリプト識別プログラム（図示せず）をCPUc1が読み出してかつ実行することにより、機能する仮想的な機能単位である。

【0069】なお、プログラム格納部21は、具体的に言うと、次のような基本機能提供プログラムD22を有する。すなわち、プログラム格納部21は、例えば、

- ①新規にワークフロー処理を開始させる機能を実現する基本機能提供プログラムD22と、
- ②実行中のワークフロー処理の進行状況を確認する機能を実現する基本機能提供プログラムD22と、
- ③ワークフロー処理を中止する機能を実現する基本機能提供プログラムD22と、
- ④ワークフロー処理における全てのアクティビティの終了を確認する機能を実現する基本機能提供プログラムD22と、
- ⑤中止または終了したワークフロー処理に関する情報を破棄する機能を実現する基本機能提供プログラムD22と、
- ⑥処理者毎に設けられたワークリストにワークアイテムを付加する機能を実現する基本機能提供プログラムD22と、
- ⑦ワークリストにおけるワークアイテムの削除によってアクティビティの終了を認識する機能を実現する基本機能提供プログラムD22とを有する。

【0070】なお、図4に示すように、ここでは階層構造を有するワークフローエンジンのクラスがホスト計算機h1、h2およびh3に分散して存在しているため、プログラム格納部21は、⑧ワークフローエンジンサブクラスが他のワークフローエンジンと相互に通信するための通信機能を実現する基本機能提供プログラムD22を有するのが好ましい。また、プログラム格納部21は、⑨人間による処理を必要としないアクティビティについて、起動アプリケーションによってアクティビティを自動処理する基本機能提供プログラムD22を有しても良い。また、図4に示すように、プログラム格納部21は、例えばホスト計算機h1の記憶装置m1のデータ領域として設けられる。

【0071】また、Java言語によるOOPを行うには、プログラム格納部21が、ワークフローエンジンのスーパークラス等のワークフロー処理に利用できるクラスを定義したクラスライブラリ52を有するのが望ましい。図3では、クラスライブラリD52はワークフローエンジンスーパークラスコードD27s (DWorkflowEngine) およびその他のスーパークラスコードD27t (LoanApp2) を含む。

【0072】ワークフローエンジンコード生成手段F19は、記憶装置m1のプログラム格納部21から上述のような①～⑨の基本機能提供プログラムD22、記憶装置m1に蓄積されたオブジェクト定義スクリプトD43aおよびフロー定義スクリプトD43bを読み出したのち、読み出された①～⑨の基本機能提供プログラムD22を組み合わせ、ワークフローエンジンサブクラスF27a (LoanTeller) およびF27b (LoanDecision) として機能するワークフローエンジンサブクラスコードD27a (LoanTeller) およびD27b (LoanDecision) を生成する。

【0073】このとき、ワークフローエンジンコード生成手段F19は、ワークフロー処理に関するオブジェクト定義スクリプトD43aおよびフロー定義スクリプトD43bを解釈することにより、当該ワークフロー処理に必要な基本機能のみを有するワークフローエンジンサブクラスコードD27a (LoanTeller) およびD27b (LoanDecision) を生成する。すなわち、ワークフローエンジンコード生成手段F19は、オブジェクト定義スクリプトD43aおよびフロー定義スクリプトD43bで定義されたワークフロー処理に必要な基本機能のみを読み込みかつ不必要な機能を読み込まない。例えば、スクリプト識別手段F45から読み出されたオブジェクト定義スクリプトD43aおよびプログラム格納部21から読み出された①～⑨の基本機能の幾つかを組み合わせ、ワークフローエンジンサブクラスコードD27a

(LoanTeller) およびD27b (LoanDecision) が生成される。そのため、このワークフローエンジンサブクラスコードD27a (loanTeller) およびD27b (LoanDecision) は、必要最低限の構成のみを有し、従って占有メモリが小さくなる。具体的に言うと、例えば、従来のワークフローエンジンでは任意のステップ数のアクティビティが処理できなければならないが、実施の形態のワークフローエンジンは、2つのアクティビティが処理できればよいから、余分なクラスやメソッドの定義を省略したコードとなる。

【0074】オブジェクト定義コード生成手段F15は、記憶装置m1のプログラム格納部21から上述のような基本機能提供プログラムD22と、記憶装置m1に蓄積されたオブジェクト定義スクリプトD43aとを読み出したのち、読み出された基本機能提供プログラムD22およびオブジェクト定義スクリプトD43aを組み合わせ、オブジェクト定義コードD47aを生成する。

【0075】フロー定義コード生成手段F17は、記憶装置m1のプログラム格納部21から上述のような基本機能提供プログラムD22と、記憶装置m1に蓄積されたフロー定義スクリプトD43bとを読み出したのち、読み出された基本機能提供プログラムD22およびフロー定義スクリプトD43bを組み合わせ、フロー定義コードD47bを生成する。

【0076】図6および図7は、オブジェクト定義コード生成手段F15およびフロー定義コード生成手段F17によってJava言語によるソースプログラムコードとして生成されたオブジェクト定義コードD47aおよびフロー定義コードD47bの一部を示した図である。

【0077】図6に示すソースプログラムコードのうち1～2行目のコードは、ワークフローエンジンのスーパークラスコードD27s (DWorkflowEngine) に対するサブクラスコードD27a (loanDecision) を定義する。3～7行目のコードは、ワークフローエンジンサブクラスコードD27b (loanDecision) について例外処

理を宣言している。8行目～11行目のコードは、ワークフローエンジンサブクラスコードD27b (loanDecision) における処理内容を示すメソッドdecideLoanを定義する。また、12～24行目のコードは、ワークフローエンジンサブクラスコードD27b (loanDecision) を開始したとき最初に実行されるメソッドが定義される。

【0078】図7に示すソースプログラムコードのうち1～2行目のコードによって、ワークフロー処理に必要なその他のスーパークラスコードD27t (LoanApp2) のサブクラスLoanFlow2 が定義される。3～7行目のコードは、その他のスーパークラスコードD27t (LoanFlow2) の初期化および名前の定義を行う。8～26行目のコードは、loanFlow2 に用いるワークフローエンジンサブクラスコードD27a (loanTeller) およびD27b (loanDecision) を登録する。27～31行目のコードは、ワークフローエンジンサブクラスコードD27a (loanTeller) およびD27b (loanDecision) の名前のテーブルを作成する。32～34行目のコードは、ワークフロー処理の名前のテーブルを作成する。35～37行目のコードは、ワークフロー処理のタイムリミットテーブルを作成する。38～40行目のコードは、ワークフロー処理の終了を判定するテーブルを作成する。

【0079】また、実際にワークフロー処理を行うとき、図3に示すワークフローエンジンサブクラスF27a (loanTeller) およびF27b (LoanDecision) が駆動して、オブジェクト定義コードD47a およびフロー定義コードD47bのクラスやメソッドの参照を行うことにより、ワークフロー処理を制御する。

【0080】また、実際にワークフロー処理を行うとき、オブジェクト定義コードD47a およびフロー定義コードD47bには、プロセスに関する定義が記述されていなければならない。これらの定義コードD47a およびD47bは、一つのステップで生成しても良いし、或いは複数ステップに分けて生成しても良い。例えば、コードジェネレータF28では基本的な定義コードのみを生成し、基本的な定義コードに具体的な定義コードを追加することにより、定義コードを完成させても良い。言い換えれば、複数ステップに分けてオブジェクト定義コードD47aやフロー定義コードD47bを生成する場合、まずオブジェクト定義基本コードとして各アクティビティに共通の処理内容を定義するコードを生成し、および、オブジェクト定義拡張コードとして個々のアクティビティに固有の処理内容を定義するコードをオブジェクト定義基本コードに追加することにより、オブジェクト定義コードを生成しても良い。具体的に言うと、図6のオブジェクト定義コードD47aは、8～9行目のブロックにオブジェクト定義拡張コードとして、個々のアクティビティに固有のメソッドが追加されると、オブジェクト定義コードとして完成する。このように、オブ

ジェクト定義コード生成手段F15がオブジェクト定義基本コードを生成したのち、オブジェクト定義基本コードにオブジェクト定義拡張コードを付加するのが好ましい。それにより、個々のオブジェクト定義コードが簡潔となる。

【0081】なお、説明の都合上、仮想的な各機能単位である識別手段、ワークフローエンジンコード生成手段、オブジェクト定義コード生成手段、フロー定義コード生成手段を特定の同一ホスト計算機のCPUに存在させたが、これら各機能単位は任意のホスト計算機のCPUに存在できる。また、プログラム格納部21、オブジェクト定義コードD47a、フロー定義コードD47bも任意のホスト計算機の記憶装置に存在して良い。

【0082】以上のような手順で、プロセス定義スクリプトでワークフロー処理のプロセスを定義することにより、ワークフローエンジンサブクラスコードD27a (LoanTeller) およびD27b (LoanDecision) と、オブジェクト定義コードD47aと、フロー定義コードD47bとが生成できる。ただし、ワークフロー処理の開始の際には、ホスト計算機h1で生成されたワークフローエンジンサブクラスコードD27a (LoanTeller) およびD27b (LoanDecision) を、それぞれホスト計算機h2およびホスト計算機h3に分散させる。

【0083】従来のワークフロー管理システムと同様に、図3に示す各ワークフローエンジンサブクラスF27a (loanTeller) およびF27b (LoanDecision) と同一ホスト計算機には、各々ワークリストD53aおよびD53bと、ワークリストハンドラF55aおよびF55bとが設けられている。ワークリストD53aおよびD53bは、個々の処理者31aおよび31b毎に設けられる。

【0084】各ホスト計算機h2およびh3では次のような処理が行われる。すなわち、まず、各処理者31aは、ワークフローエンジンサブクラスF27a (loanTeller) によってワークリストD53aに示されたワークアイテムを参照する。それにより、処理者31aは処理すべきアクティビティを認識する。そして、処理者31aはクライアントアプリケーションD57等を用いてアクティビティを実行する。アクティビティに含まれる作業が終了すると、ワークリストハンドラF55aはワークリストD53aからワークアイテムを削除する。それにより、ワークフローエンジンサブクラスF27a (loanTeller) は、アクティビティの終了を検出する。その後、ワークフローエンジンF27a (loanTeller) は、アクティビティで処理されたデータを次のワークフローエンジンF27b (loanDecision) に送付する。

【0085】そして、各処理者31bは、ワークフローエンジンサブクラスF27b (loanDecision) によってワークリストD53bに示されたワークアイテムを参照する。処理者31bは処理すべきアクティビティを認識

する。そして、処理者31bはクライアントアプリケーションD57等を用いてアクティビティを実行する。アクティビティが終了すると、処理者31bはワークリストハンドラF55bを操作してワークリストD53bからワークアイテムを削除する。それにより、ワークフローエンジンサブクラスF27b (loanDecision) は、アクティビティの終了を検出する。同時に、ワークフローエンジンF27b (loanDecision) は、ワークフロー処理の全てのアクティビティの終了を検出して、例えば処理者31a等にワークフロー処理の終了および結果を通知する。

【0086】続いて、上述のワークフロー管理システムを一適用例としてワークフロー処理のプロセスを変更する方法につき説明する。

【0087】実施の形態のワークフロー管理システムのプロセスの変更は、オブジェクト定義コードD25a

(D47a) 或いはフロー定義コードD25b (D47b) を書き換えて行う。したがって、従来技術のワークフロー管理システムのようにプロセス定義データそのものを書き換える必要がない。すなわち、実施の形態のワークフロー管理システムによれば、各アクティビティの処理内容のみを変更したい場合にはオブジェクト定義コードD25aのみを修正し、アクティビティの順序のみを変更したい場合にはフロー定義コードD25bのみを修正すればよい。

【0088】具体的に言うと、例えば図7のフロー定義コードに従うワークフロー処理において、処理者を変更したい場合には、フロー定義コードに記述されたワークフローエンジンサブクラス (loanDecision) を別のワークフローエンジンサブクラスに名称の変更をすればよい。また、例えば図6のオブジェクト定義コードに従うワークフロー処理において、ワークフローエンジンサブクラス (loanDecision) の処理内容を変更したい場合には、オブジェクト定義コードの処理内容を示すメソッド decideLoan を修正すればよい。このように、オブジェクト定義コード或いはフロー定義コードの一方の修正が他方の修正に影響しないため、プロセスは容易に変更できる。

【0089】オブジェクト定義コード或いはフロー定義コードを修正するには、前述のワークフロー管理システムが、オブジェクト定義コードおよび前述のフロー定義コードを高水準言語のコードの状態で作成する定義コード保存手段 (図示していない。) を更に具えるのが望ましい。

【0090】図6および図7に示すオブジェクト定義コードおよびフロー定義コードは、ワークフロー処理を開始する前に生成される。これらの定義コードはこの実施の形態ではソースプログラムである。これらの定義コードはコンパイルされることにより、動作可能なオブジェクトプログラムとなる。例えば、Java言語ソースプ

ログラムをコンパイルすると、ソースプログラムはプラットフォームに依存せずに動作するバイトコードに変換されて実行型プログラムとなる。コンパイルしたバイトコードデータそのものを書き換えるのは困難であるから、コンパイルする前の状態すなわち例えばJava言語のソースプログラムコードを保存しておく。それにより、各アクティビティにおける処理内容或いは処理順序を変更する際、Java言語のソースプログラムコードの一部を修正することにより、新たなオブジェクト定義コード或いはフロー定義コードが作成できる。新たな定義コードを、コンパイルしてバイトコードに変換したのち修正前のバイトコードデータと置換すれば、プロセスに関する定義が修正できる。

【0091】(第2の実施の形態) 以下の第2の実施の形態では、第1の実施の形態のワークフロー管理システムに宛名変換手段を設けた形態につき説明する。ただし、ここでは特に階層構造を有するワークフローエンジンを設けている形態につき説明する。第1の実施の形態にて説明したワークフロー管理システムでは、アクティビティの処理順序はフロー定義コードに静的に定義される。すなわち、アクティビティの処理者を示すコードには具体的な宛先が示されており、人事異動等による宛先の変更すなわちアクティビティの順序の変更があった場合、フロー定義コードを書き換える必要がある。

【0092】ところで、文献I等に開示されている宛名変換手段を用いると、動的に処理順序を定義できるため、プロセス定義データを修正する必要がなくなることが知られている。しかしながら、文献Iに開示されるワークフロー管理システムの場合、ワークフローエンジンはサーバで、また、宛名変換手段はワークフローエンジンと同一のサーバで動作する。そのため、宛名変換手段を含むサーバの負担が大きくなる。また、実際にサーバがダウンしたときには、ワークフロー処理が中断されてしまうという問題があった。

【0093】図8は、第2の実施の形態のワークフロー管理システムを動作させるワークフロー管理システムの各機能単位を模式的かつ概略的に示すブロック図である。ただし、図8では図4と同一の機能単位については示しておらず、図4と異なる点のみすなわちワークフローエンジンサブクラスF27a、F27bおよび宛名変換手段F67が示される。

【0094】図8に示すように、第2の実施の形態のワークフロー管理システムは、宛名変換手段F63を有する。ここで、ホスト計算機h4のCPUc4が記憶装置m4の宛名変換プログラムD63を読み出しかつ実行することにより、宛名変換手段F63は仮想的な機能手段としてCPUc4で動作する。

【0095】図8に示すワークフローエンジンは、OOPによってプログラミングされており、スーパークラスから複数のサブクラスF27aおよびF27bに機能を

10

20

30

40

50

継承するクラス構造を有する。そして、個々のワークフローエンジンサブクラスF 2 7 aおよびF 2 7 bは、各処理者毎に分散して存在するため、互いにアクセスするための各通信機能F 6 7 aおよびF 6 7 bを有する。

【0096】そして、この第2の実施の形態の宛名変換手段F 6 3は、特に、通信機能F 6 7 tを備える点を特徴とする。よって、宛名変換手段F 6 3と、複数のワークフローエンジンサブクラスF 2 7 aおよびF 2 7 bとは通信によって互いにアクセスできる。

【0097】宛名変換手段F 6 3は、周知のごとく、課長や部長など組織内の役職等を示す一般的宛先名を、処理者の名前等の具体的宛先名に変換する機能モジュールである。また、この宛名変換手段F 6 3は、従来構成と同様に組織／役割データD 6 5を参照する。例えば備品購入の決裁のワークフロー処理を行う場合、決裁ルートは役職で決定されることが多いから、予め宛先名を役職で指定する。そのため、決裁ルートに沿う役職にある処理者が人事異動等によって他の処理者になっても動的に宛先が判断されるため、決裁ルートすなわち宛先名を変更する必要が無い。

【0098】ここで図8を参照して、宛名変換手段6 3によって一般的宛先名から具体的宛先名を判断する流れについて説明する。

【0099】ワークフローエンジンサブクラスF 2 7 aは、フロー定義コード中に定義されたアクティビティの処理者が一般的な宛先名で記述されている場合、先ず、そのワークフロー処理を開始させたワークフローエンジンサブクラス名すなわち処理者を調べる。次に、ワークフローエンジンサブクラスF 2 7 aは、各通信機能F 6 7 aおよびF 6 7 tを介して宛名変換手段F 6 3とネットワーク通信することにより、一般的宛先名および開始者名を宛名変換手段F 6 3に送信する。

【0100】このとき、宛名変換手段6 3は、実行中のワークフロー処理の開始者名から開始者の組織内の役割（所属部署や役職等）を認識すると共に、この開始者の役割および一般的宛先名から次のアクティビティを担当する処理者名を調べる。そして、次の処理者名からワークフローエンジンサブクラス名を得て、次のアクティビティを処理すべきワークフローエンジンサブクラス名を、ワークフローエンジンサブクラスF 2 7 aに送信する。そして、この後、次のアクティビティを処理すべきワークフローエンジンサブクラスF 2 7 bが起動する。

【0101】以上のように、第2の実施の形態では、図8に示すようにワークフローエンジンが階層構造をなす複数のワークフローエンジンサブクラスF 2 7 aおよびF 2 7 bとして異なるホスト計算機h 3およびh 4に分散しており、および、宛名変換手段F 6 3が通信機能を有する。すなわち、ホスト計算機h 2ではワークフローエンジンサブクラスF 2 7 aが動作し、ホスト計算機h 3ではワークフローエンジンサブクラスF 2 7 bが動作

し、および、ホスト計算機h 4では宛名変換手段F 6 3が動作する。このように、宛名変換手段F 6 3と、ワークフローエンジンサブクラスF 2 7 aおよびF 2 7 bとは異なるホスト計算機に存在しても、宛名変換手段F 6 3が通信機能F 6 7 tを有するため、宛名変換手段F 6 3はワークフローエンジンサブクラスF 2 7 aおよびF 2 7 bと互いに通信できる。したがって、宛名変換手段F 6 3を任意のホスト計算機に移動させることができる。すなわち、従来のように宛名変換手段F 6 3をワークフローエンジンと同一ホストに設ける必要がない。

【0102】そのため、宛名変換手段を含むホスト計算機の負担を小さくしシステムダウンの可能性を低減させることができる。また、宛名変換手段を含むホスト計算機がシステムダウンしたときでも、別のホスト計算機に宛名変換手段を移動することによって、ワークフロー処理を続行できる。

【0103】

【発明の効果】上述した説明から明らかなように、この発明のワークフロー管理システムによれば、ワークフロー処理のプロセスを定義をする際、起票者が入力したプロセス定義情報は、識別手段によってフロー定義に関する部分およびオブジェクト定義に関する部分に分離されたのち、フロー定義コード生成手段およびオブジェクト定義コード生成手段によってコード化される。そのため、ワークフロー処理を行うとき、各アクティビティにおける処理内容およびアクティビティの順序を変更する場合であっても、プロセス定義データそのものを修正する必要がない。言い換えれば、各アクティビティにおける処理内容のみを変更したい場合にはオブジェクト定義コードのみを修正すれば良く、アクティビティの順序（アクティビティを行う処理者の変更）を変更したい場合にはフロー定義コードのみを修正すればよい。したがって、プロセスを能率的にかつ容易に変更することができる。

【0104】また、この発明のコードジェネレータによれば、識別手段が、プロセス定義情報をオブジェクト定義情報およびフロー定義情報に分離させたのち、オブジェクト定義情報からオブジェクト定義コードを生成しかつフロー定義情報からフロー定義コードを生成する。そのため、各工程における処理内容のみを変更したい場合にはオブジェクト定義コードを修正し、工程の処理順序のみを変更したい場合にはフロー定義コードを修正すればよい。よって、プロセスを能率的にかつ容易に変更することができる。

【0105】また、このワークフロー処理のプロセスの変更方法の発明によれば、プロセス定義情報から、オブジェクト定義コードおよびフロー定義コードを生成するワークフロー処理のプロセスを変更するのに、各工程の処理内容のみを変更する場合にはオブジェクト定義コードのみを修正し、工程の処理順序のみを変更する場合に

はフロー定義コードのみを修正する。したがって、このプロセスの変更方法によれば、各工程における処理内容のみを変更したい場合若しくは工程の処理順序のみを変更したい場合のいずれにも、互いに独立に存在するオブジェクト定義コード或いはフロー定義コードの一方を修正することにより対応できる。よって、プロセスを能率的にかつ容易に変更できる。

【図面の簡単な説明】

【図1】第1の実施の形態のワークフロー管理システムの機能単位の相互関係を概略的に示すブロック図である。

【図2】図1の各機能単位を1台のホスト計算機で動作させた場合を例として、各機能単位によるハードウェア資源の利用形態を概略的に示す図である。

【図3】図1のワークフロー管理システムの具体的な機能構成を概略的に示した図である。

【図4】図3のワークフロー管理システムの各機能単位を動作させるネットワークシステムの一例を示す図である。

【図5】具体的なワークフロー処理に対するプロセス定義スクリプトの一例である。

【図6】ソースプログラムコードとして生成されたオブジェクト定義コードの一部を示した図である。

【図7】ソースプログラムコードとして生成されたフロー定義コードの一部を示した図である。

【図8】第2の実施の形態のワークフロー管理システムを動作させるネットワークシステムの各機能単位を模式的かつ概略的に示すブロック図である。

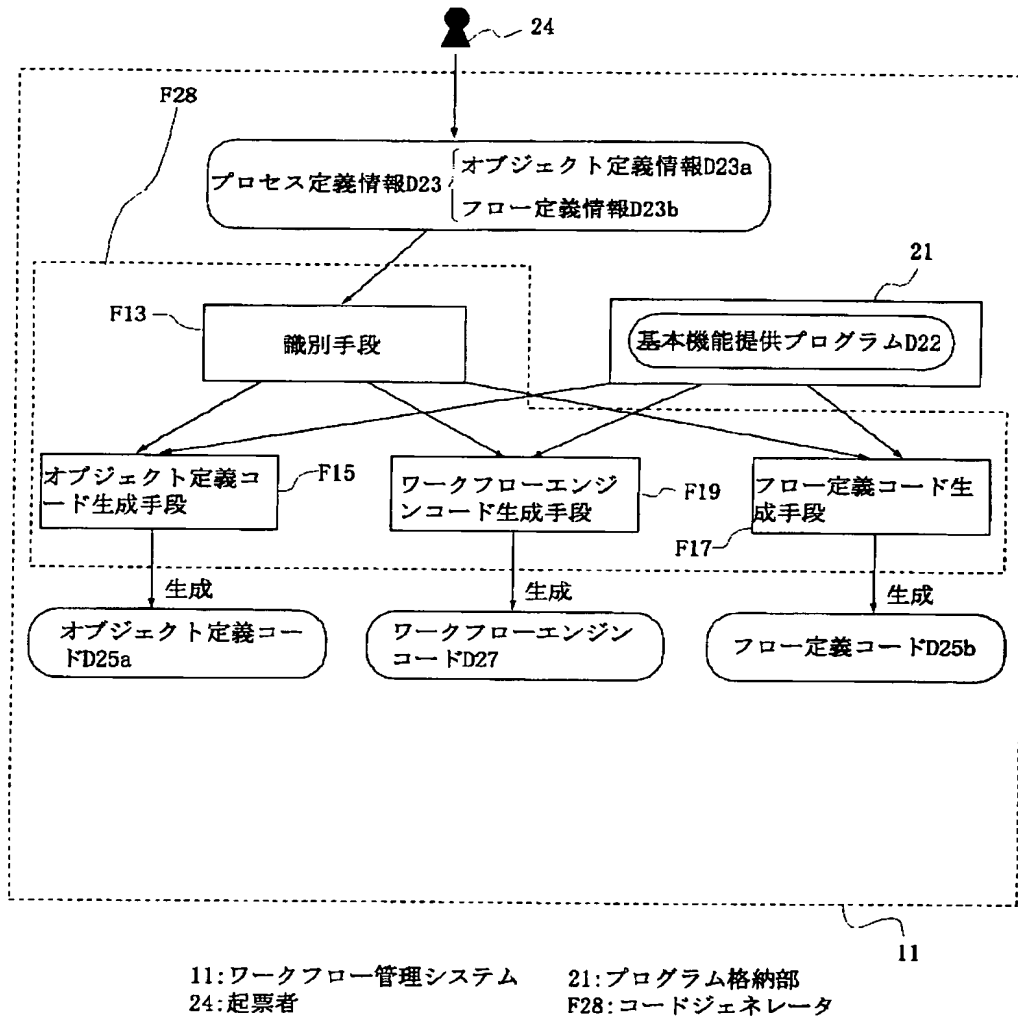
【図9】従来のワークフロー管理システムの構成を示すブロック図である。

【符号の説明】

11、41：ワークフロー管理システム  
F13：識別手段  
D13：識別プログラム

F15：オブジェクト定義コード生成手段  
D15：オブジェクト定義コード生成プログラム  
F17：フロー定義コード生成手段  
D17：フロー定義コード生成プログラム  
F19：ワークフローエンジンコード生成手段  
D19：ワークフローエンジンコード生成プログラム  
21：プログラム格納部  
D22：基本機能提供プログラム  
D23：プロセス定義情報  
D23a：オブジェクト定義情報  
D23b：フロー定義情報  
24：起票者  
D25a、D47a：オブジェクト定義コード  
D25b、D47b：フロー定義コード  
D27：ワークフローエンジンコード  
F27a、F27b：ワークフローエンジンサブクラス  
D27a、D27b：ワークフローエンジンサブクラスコード  
D27s：ワークフローエンジンスーパークラスコード  
D27t：その他のスーパークラスコード  
F28：コードジェネレータ  
31a、31b：処理者  
D43：プロセス定義スクリプト  
D43a：オブジェクト定義スクリプト  
D43b：フロー定義スクリプト  
F45：スクリプト識別手段  
D52：クラスライブラリ  
D53a、D53b：ワークリスト  
F55a、F55b：ワークリストハンドラ  
D57：クライアントアプリケーション  
F63：宛名変換手段  
D63：宛名変換プログラム  
D65：組織／役割データ  
F67a、F67b、F67t：通信機能

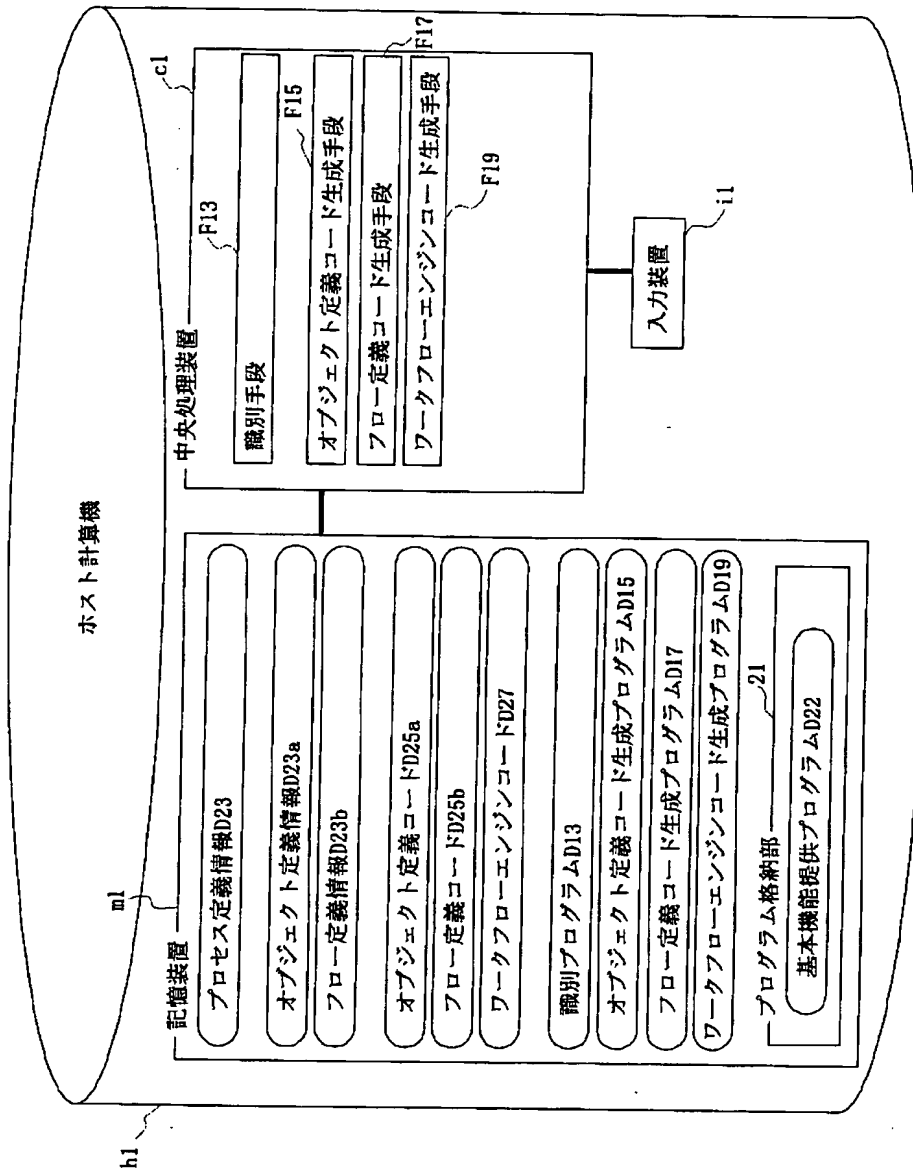
【図1】



第1の実施の形態のワークフロー管理システム

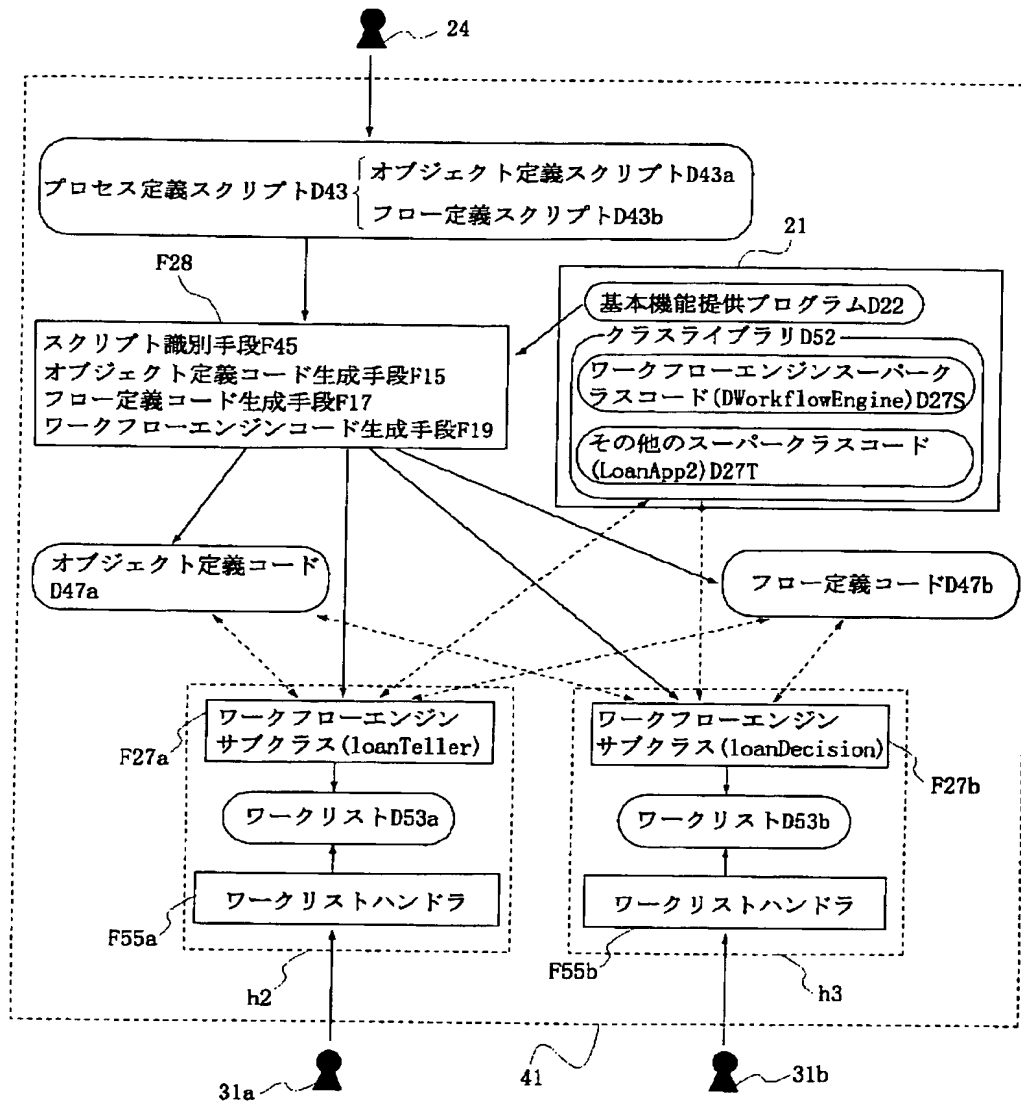


【図2】



ハードウェア資源の利用の形態の説明に供する図

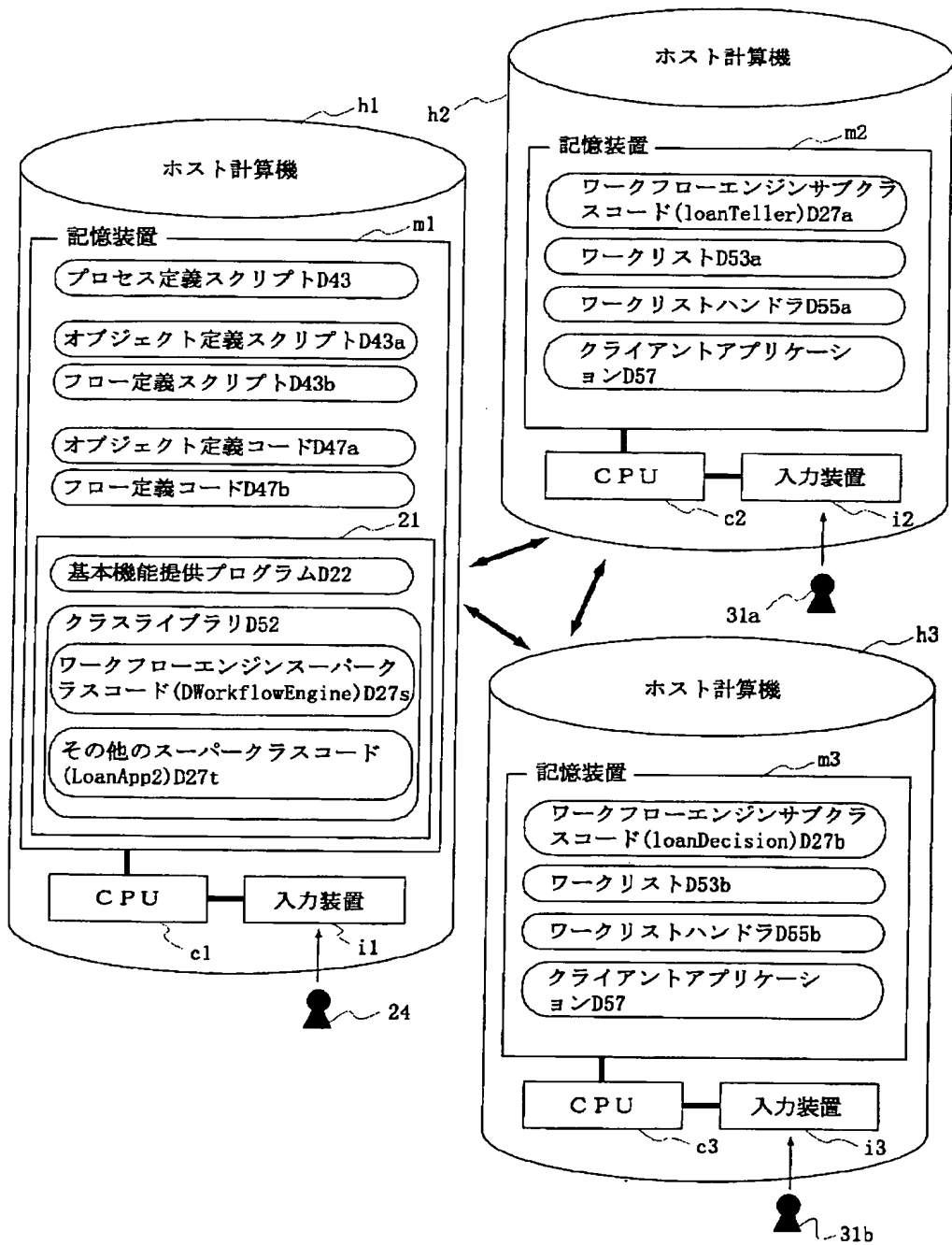
【図3】



h2, h3: ホスト計算機    31a, 31b: 処理者    41: ワークフロー管理システム

第1の実施の形態のワークフロー管理システムの具体的な機能構成

【図4】



ワークフロー管理システムを動作させるネットワークシステムの構成例

【図5】

```
aspect Flow relates LoanApp2
  engines {
    loanTeller — "loanTeller";
    loanDecision — "loanDecision";
  }
  flow {
    loanTeller;
    loanDecision;
    loanTeller;
  }
  do (loanTeller) {
    %E.start(%W);
  }
  do (loanDecision) {
    %E.decideLoan(%W);
  }
}
```

} D43b

} D43a

プロセス定義スクリプトの一例

【図6】

```
1: public class LoanDecison extends DWorkflowEngine
2: {
3:     public LoanDecision(String name)
4:         throws java.rmi.RemoteException
5:     {
6:         super(name);
7:     }
8:     public void decideLoan(MessageObject[] mos)
9:     {
10:         System.out.println("invoke decideLoan()");
11:     }
12:     public static void main(String arg[])
13:         throws Exception
14:     {
15:         try {
16:             System.out.println("LoanDecision boot");
17:             LoanDecision de = new LoanDecision("LoanDecision");
18:             bind(de);
19:             de.act();
20:         } catch(Exception e) {
21:             e.printStackTrace();
22:         }
23:     }
24: }
```

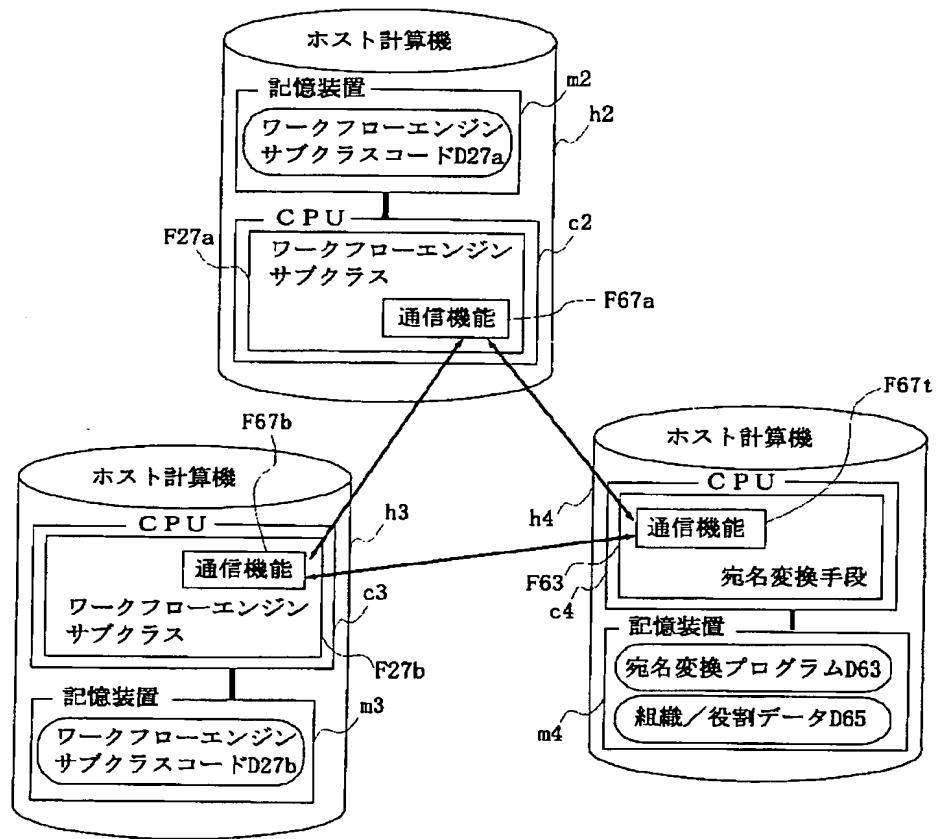
J a v a ソースプログラムコードによるオブジェクト定義コードの一部

【図7】

```
1: public class LoanFlow2 extends LoanApp2
2: {
3:     public LoanFlow2(WorkItems workitems)
4:     {
5:         super(workitems);
6:         flowName = "LoanFlow2";
7:     }
8:     public void makeFlowTree()
9:     {
10:         DEItem deItem = null;
11:         NFItem nfItem = null;
12:         BasicFlowElement flow0 = null;
13:
14:         // DE:loanTeller
15:         deItem = new DEItem();
16:         deItem.name = "loanTeller";
17:         link(flow0, deItem);
18:         deVSTreeTable.put(deItem.name, deItem);
19:
20:         // DE:loanDecision
21:         deItem = new DEItem();
22:         deItem.name = "loanDecision";
23:         link(flow0, deItem);
24:         deVSTreetable.put(deItem.name, deItem);
25:     }
26:     public void makeDENameTable()
27:     {
28:         deNameTable.put("loanDecision", "loanDecision");
29:         deNameTable.put("loanTeller", "loanTeller");
30:     }
31:     public void makeFlowNameTable()
32:     {
33:     }
34:     public void makeTimeLimitTable()
35:     {
36:     }
37:     public void makeAcceptTable()
38:     {
39:     }
40: }
41: }
```

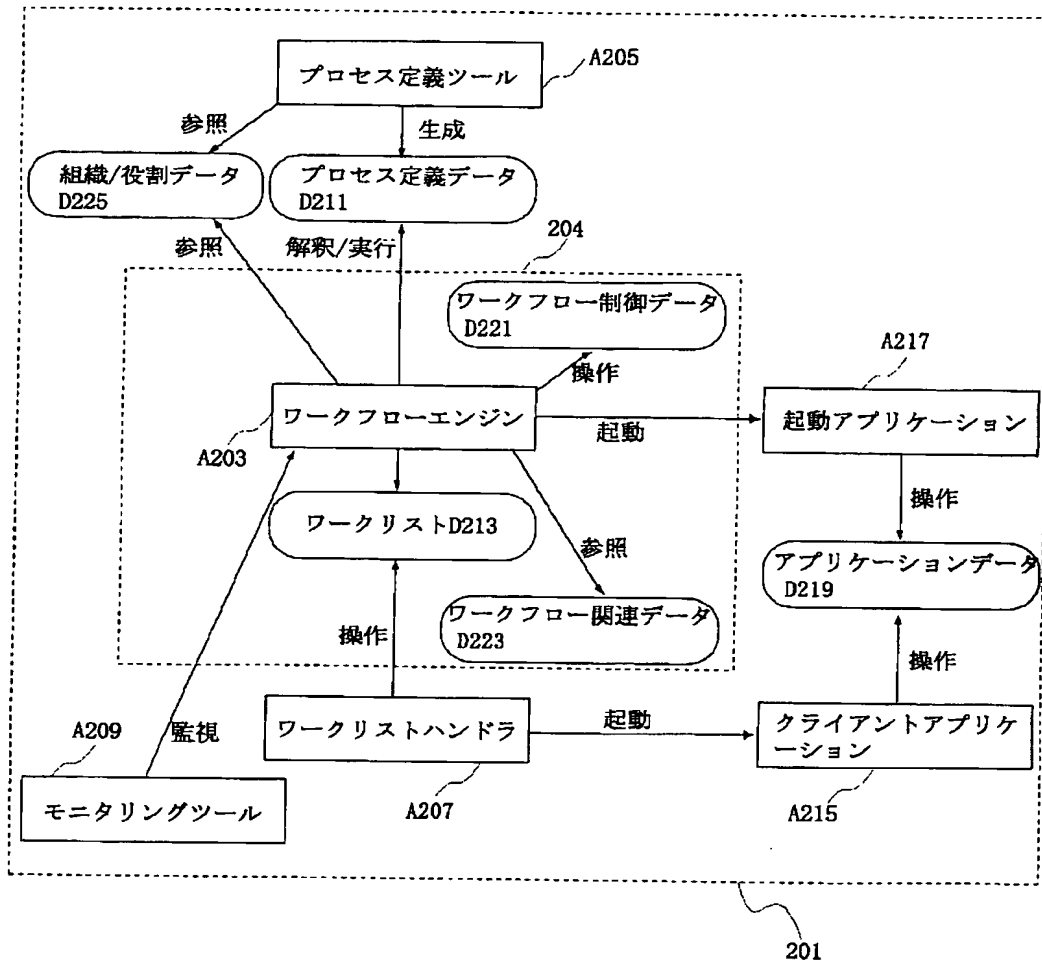
J a v a ソースプログラムコードによるフロー定義コードの一部

【図8】



第2の実施の形態のワークフロー管理システムを動作させるネットワークシステムの構成例

【図9】



従来ワークフロー管理システム



**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**